Universidad Nacional de Asunción
Facultad Politécnica

Doctorado en Ciencias de la Computación

Programas de Evaluación en Grupos Topológicos

Pedro Alberto Villalba Sosa

Proyecto de Tesis Doctoral presentado en conformidad a los requisitos para obtener el título de Doctor en Ciencias de la Computación

San Lorenzo - Paraguay
Diciembre - 2022

Universidad Nacional de Asunción
Facultad Politécnica

Doctorado en Ciencias de la Computación

Programas de Evaluación en Grupos Topológicos

Pedro Alberto Villalba Sosa

San Lorenzo - Paraguay
Diciembre - 2022

Universidad Nacional de Asunción
Facultad Politécnica

Doctorado en Ciencias de la Computación

Programas de Evaluación en Grupos Topológicos

Pedro Alberto Villalba Sosa

Orientador: Prof. Dr. Marcos Daniel Villagra Riquelme

Proyecto de Tesis Doctoral presentado en conformidad a los
requisitos para obtener el título de Doctor en Ciencias de la Computación

San Lorenzo - Paraguay
Diciembre - 2022

Programas de Evaluación
en Grupos Topológicos

Trabajo de fin de doctorado para la obtención del título de Doctor en Ciencias de la Computación aprobado en representación de la Facultad Politécnica de la Universidad Nacional de Asunción, por el Tribunal Examinador constituido por los siguientes profesores.

Prof. Dr. Sebastian Alberto Grillo

Prof. Dr. Jorge Daniel Mello Román

Prof. Dr. Inocencio Esteban Ortiz Samudio

Marcos Daniel Villagra Riquelme
Orientador

Horacio Andrés Legal Ayala
Director del Programa del postgrado

Acta N° 02-2022-13

Fecha de aprobación: 19 de diciembre de 2022

Dedicado a mis padres, hermanos
esposa e hijos.

AGRADECIMIENTOS

Un agradecimiento muy especial al Prof. Dr. Marcos Villagra, por su inmensa paciencia y perseverencia para la realización de este trabajo.

A la Facultad Politécnica por haberme brindado esta oportunidad.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) que a través de sus diferentes convocatorias, proyectos de investigación y programas de incentivo ha contribuido a la realización de este programa y en particular, su aporte invaluable para que pueda llegar a la meta propuesta.

Por último, a mi familia, por el apoyo incondicional en esta larga odisea del doctorado.

A todos, ¡muchas gracias!

Programas de Evaluación en Grupos Topológicos

Autor: Pedro Alberto Villalba Sosa

Orientador: Marcos Daniel Villagra Riquelme

RESUMEN

Los programas de evaluación se utilizan en los estudios de algoritmos para estructuras algebraicas y establecen los fundamentos teóricos del álgebra computacional. En este trabajo estudiamos y analizamos la complejidad algebraica de computar conjuntos utilizando programas de evaluación construidos sobre grupos. En particular, nos interesa saber como utilizando ciertas familias de grupos podemos computar otras estructuras algebraicas mediante programas de evaluación. Para ese fin, introducimos una idea de simulación de programas de evaluación con entradas, el cual nos permite determinar conexiones entre las diferentes complejidades de computar conjuntos respecto a diferentes familias de grupos. Demostramos además una relación entre la complejidad de programas de evaluación sobre grupos conmutativos y programas de evaluación sobre grupos no conmutativos.

Palabras Clave: complejidad computacional, grupos, programas de evaluación, simulación.

Straight-line Programs in Topological Groups

Author: Pedro Alberto Villalba Sosa

Advisor: Marcos Daniel Villagra Riquelme

ABSTRACT

Straight-line programs are important in the study of computer algebra. In this work, we investigate the algebraic complexity of sets presented by straight-line programs over groups. In particular, we are interested in how subsets of certain families of groups can be computed by other families of groups. We introduce an idea of simulation by straight-line programs which can be used to relate complexity measures of subsets of groups. As a main result of this paper, we prove that for any straight-line program computing a set over an conmutative group, there exists a shorter straight-line program computing the same set (up to isomorphism) over a nonconmutative group.

Keywords: computational complexity, groups, straight-line programs, simulation.

ÍNDICE

Página

LISTA DE SÍMBOLOS

$g \in G$	—	El elemento g pertenece al conjunto G.
$g \notin S$	—	El elemento g no pertenece al conjunto S.
$F \subseteq G$	—	F es un subconjunto de G o F está contenido en G.
$F \supseteq A$	—	F contiene al conjunto A.
$H \leqslant G$	—	H es un subgrupo de G
$G = \langle S \rangle$	—	S es un conjunto generador de G
Γ_G	—	Programa de evaluación Γ en G.
$\Gamma_G \prec \Lambda_{G'}$	—	$\Lambda_{G'}$ simula a Γ_G.
$<$	—	Menor entre números.
$>$	—	Mayor entre números.
$\leq$	—	Menor o igual entre números.
$\geq$	—	Mayor o igual entre números.
$\mathbb{Z}$	—	El conjunto de los números enteros.
$f : G \to H$	—	Función f entre los conjuntos G y H.
$H \cap K$	—	La intersección del conjunto H y del conjunto K.
$\bigcap_{G \in \mathcal{G}} G$	—	La intersección del todos los conjuntos $G \in \mathcal{G}$.
$H \cup K$	—	La unión del conjunto H y del conjunto K.
$\bigcup_{U \in \mathcal{U}} U$	—	La unión de todos los conjuntos $U \in \mathcal{U}$.
$\mathbb{Z}^+$	—	El conjunto de los números enteros positivos.
S^n	—	El producto cartesiana de S consigo mismo n veces.
D_6	—	Grupo Diedral de orden 12 de las simetrías de un polígono de 6 lados.
$:=$	—	Se define como.
S_6	—	Grupo simétrico o de las permutaciones de 6 elementos.
(S^n, a)	—	La entrada $a \in S^n$ para un programa de evaluación con entradas de longitud n.
$L_G(F, S^n)$	—	El costo de evaluación de F respecto S^n.
$\lvert F \rvert$	—	El cardinal del conjunto F.
$\mathcal{L}_{\mathcal{G}}(F, n)$	—	Complejidad de evaluación de un conjunto F respecto a la familia $\mathcal{G}$.
$\hat{F}_{G'}$	—	Extensión de F en G'.
$\varphi \circ \phi$	—	Composición de las funciones φ y ϕ.
τ	—	Topología sobre algún conjunto.
$\mathscr{B}$	—	Base para alguna topología.
$\mathscr{S}$	—	Subbase para alguna topología.
$\mathscr{P}(X)$	—	Partes de X o colección de todos los subconjuntos de X.
τ_A	—	Topología relativa a un conjunto A.

$\mid$	—	Tal que.
$\forall$	—	Para todo o para cada.
$\exists$	—	Existe.
$X \times Y$	—	Producto cartesiano de X e Y.
$\prod_{i=1}^{n} X_i$	—	Producto cartesiano de los conjuntos X_i.
$\mathscr{C}_X(A)$	—	El complemento del conjunto A respecto al conjunto X.
$\emptyset$	—	El conjunto vacío.
$\overline{A}$	—	La clausura del conjunto A.
A'	—	El conjunto derivado o conjuntos de puntos de acumulación del conjunto A.
$\wedge$	—	y (conjunción).
$f\|_A$	—	La restricción de la función en A.
$\mathbb{1}_X$	—	La función identidad en X.
$f(A)$	—	El conjunto imagen de A mediante la función f.
$A \mapsto f(A)$	—	Al conjunto A le corresponde su conjunto imagen $f(A)$.
$f^{-1}(B)$	—	El conjunto preimagen de B mediante la función f.
$B \mapsto f^{-1}(B)$	—	Al conjunto B le corresponde su conjunto preimagen $f^{-1}(B)$.
$p_j : \prod_{i=1}^{n} X_i \to X_j$	—	La función proyección en la j-ésima coordenada.
$inv : G \to G$	—	La función inversión de G.
e	—	El elemento neutro.
$\mathbb{R}^n$	—	El producto cartesiano de $\mathbb{R}$ consigo mismo n veces.
$\mathbb{C}^\times$	—	El conjunto $\mathbb{C}$ de los números complejos del que se ha eliminado el elemento cero o neutro.
$GL_n(\mathbb{R})$	—	Grupo de matrices de orden n con determinante no nulo.
γ_i	—	Funciones inducidas de un programa de evaluación.
λ_g	—	La función multiplicación por izquierda en G.
ρ_g	—	La función multiplicación por derecha en G.
μ	—	La función generada de programa de evaluación en G.

CAPÍTULO 1
INTRODUCCIÓN

1.1. Introducción

En los últimos años, hubo un resurgimiento en el estudio de la computación sobre estructuras algebraicas no conmutativas. El primer trabajo en computación no conmutativa fue el de (Nisan, 1991) quien demostró una cota inferior exponencial para calcular el determinante de una matriz utilizando programas de ramificación algebraicos no conmutativos, del inglés *algebraic branching programs*; en el mundo conmutativo, sin embargo, el determinante puede computarse con circuitos aritméticos de tamaño polinomial (Bürgisser et al., 2013). (Raz & Shpilka, 2005) presentaron el primer algoritmo no conmutativo para el problema de equivalencia de polinomios, del inglés *polynomial-identity testing*. (Arvind et al., 2016) realizaron el primer estudio de clases de complejidad no conmutativas e identificaron varias familias de polinomios no conmutativos difíciles relacionados a lenguajes formales. (Arvind, 2013) también presenta varias conexiones entre polinomios no conmutativos y autómatas.

La cota inferior exponencial para el determinante de (Nisan, 1991) es sobre un anillo libre no conmutativo para programas de ramificación, sin embargo, (Chien & Sinclair, 2004) demostraron una cota inferior exponencial para fórmulas en anillos más restringidos. Uno de los problemas abiertos más importantes en computación no conmutativa es demostrar una cota inferior exponencial para algún polinomio en circuitos aritméticos; los trabajos de (Nisan, 1991) y (Chien & Sinclair, 2004) solo se basan en modelos de circuitos restringidos.

En este trabajo, motivados por la búsqueda de técnicas para cotas inferiores en cómputos no conmutativos, estudiamos la relación entre computación conmutativa y computación no conmutativa sobre grupos utilizando programas de evaluación. Un programa de evaluación, del inglés *straight-line program*, es una secuencia finita Γ de instrucciones construida sobre una estructura algebraica.

Los programas de evaluación fueron utilizados por (Babai & Szemerédi, 1984), para estudiar la dificultad inherente de ciertos problemas computacionales en teoría de grupos. Los programas de evaluación también encontraron aplicaciones en el desarrollo de presentaciones cortas de grupos (Nies & Tent, 2017), circuitos aritméticos (Shpilka & Yehudayoff, 2010), complejidad de computaciones algebraicas (Lynch, 1980), y muchas otras aplicaciones más (Bürgisser et al., 2013).

De manera a demostrar una relación entre la computación conmutativa y la computación no conmutativa en grupos, introducimos una idea de simulación. En principio, un programa de evaluación "simula" a otro programa de evaluación si computa los mismos resultados, salvo un isomorfismo, como lo veremos más adelante.

Utilizando la idea de simulación, en el Teorema 3.1, demostramos una relación entre programas de evaluación conmutativos y no conmutativos. En particular, para cualquier programa de evaluación que compute un conjunto de resultados con los elementos de un grupo conmutativo, existe un programa de evaluación más corto que computa el mismo conjunto de resultados (salvo un isomorfismo) sobre un grupo no conmutativo.

El resultado del Teorema 3.1 va en contra de nuestra intuición. Por ejemplo, consideremos el polinomio $p(x, y) = x^2 + xy - yx + y^2$. Si el producto es conmutativo entonces $p(x, y) = x^2 + y^2$. Por lo tanto, el número de operaciones es mayor en el caso no conmutativo y menor para el caso conmutativo. Esto se refleja claramente en la separación exponencial de (Nisan, 1991), entre el determinante conmutativo y el no conmutativo.

La razón por la que el costo de los programas de evaluacón en grupos no conmutativos es más pequeño que el costo de los programas de evaluación en grupos conmutativos se debe a la falta de cancelaciones en los grupos no conmutativos, como fue ejemplificada en el párrafo anterior.

Este trabajo está estructurado de la siguiente manera. En el Capítulo 2 empezamos presentando una introducción rápida e informal a los programas de evaluación y luego mostramos algunas aplicaciones de la misma en varios trabajos de investigación que se han realizado. Aquí recalcamos la variedad de aplicaciones que tiene este modelo de cómputo que vamos a utilizar en el presente trabajo.

En el Capítulo 3 empezamos recordando algunas nociones básicas de grupo que estaremos utilizando para definir nuestro modelo de cómputo a utilizar (Sección 3.1). A partir de la Sección 3.2 comenzamos a definir de manera formal el modelo de computo que denominamos programas de evaluación en grupos y vemos también un par de ejemplos para poder comprender bien dicho concepto. En la Sección 3.3 analizamos algunas propiedades y características importantes de los programas de evaluación. Posteriormente, en la Sección 3.4 definimos las medidas con las cuales estaremos realizando el análisis de comparación entre la complejidad de evaluación conmutativa y no conmutativa, específicamente definimos el costo de evaluación de un conjunto mediante un programa de evaluación y luego la complejidad de evaluación de conjuntos respecto a una familia de grupos. En la Sección 3.5 presentamos la idea de simulación y desarrollamos una base teórica para su uso, previamente introducimos el concepto de extensión

isomorfa de un conjunto, el cual nos permitirá relacionar dos grupos para luego poder realizar la comparación de la complejidad de evaluación. En la Sección 3.6 presentamos nuestro resultado principal sobre la relación entre computación conmutativa y computación no conmutativa sobre grupos.

En el Capítulo 4 comenzamos recordando algunos conceptos básicos de espacios topológicos (Sección 4.1). En la Sección 4.2 presentamos una introducción a los grupos topológicos, vemos algunas propiedades y desarrollamos algunas demostraciones que nos servirán posteriormente. En la Sección 4.3 definimos un programa de evaluación en un grupo topológico, lo cual resulta en una familia de funciones inducidas por el programa de evaluación y luego presentamos algunos resultados interesantes que se obtienen justamente al considerar los programas de evaluación en grupos que tienen una estructura topológica específica. En la Sección 4.4 definimos en un anillo una función generada a partir de una familia de funciones inducidas por un programa de evaluación y presentamos algunas propiedades y características de la misma.

Al final, en el capítulo 5 resumimos las conclusiones obtenidas a partir de los resultados alcanzados y damos algunos problemas abiertos y lineas de investigación que se derivan de este trabajo de investigación.

CAPÍTULO 2
ESTADO DEL ARTE

Los programas de evaluación han sido utilizados para diferentes propósitos. Existen varios modelos formales de cómputos que son frecuentemente empleados por su poder de representatividad, como por ejemplo la máquina de Turing, el árbol de cómputos, el circuito aritmético, una secuencia de cómputos, entre otros.

En varios trabajos seminales se describen la utilidad de los modelos formales de cómputos, en particular nos interesa uno de ellos el cual hemos denominado aquí programas de evaluación, del ingles *straight line program*.

En este trabajo definimos los programas de evaluación sobre grupos que formalizan los cómputos paso a paso y sin ramificaciones, y analizamos su complejidad definiendo el costo de evaluación de un subconjunto de un grupo partiendo de los elementos de un conjunto generador del grupo. A partir de este conjunto de entradas, que es un subconjunto de un conjunto generador S de un grupo G, los programas de evaluación computan un subconjunto F de G por medio de las operaciones que se pueden realizar en el grupo, básicamente estas operaciones incluyen tomar un elemento del grupo, la multiplicación o el producto de elementos del grupo el cual sería la operación que hace de G un grupo y la inversión de un elemento del grupo, esto último es, dado un elemento g de G obtenemos la inversa de g en G. Al asignar costos a estas operaciones que se pueden realizar en el grupo llegamos a la noción de costo de evaluación de un conjunto F (con respecto a S y G), que es la longitud del programa de evaluación más corto que computa F en G a partir de S. La longitud de un programa de evaluación consiste en el "número" o cantidad de instrucciones que posee el mismo. Lo formalizaremos más adelante.

Existen varios artículos y trabajos de investigación que toman los programas de evaluación como modelo formal de computo para realizar sus trabajos. Algunos, por mencionar, las describimos en los párrafos siguientes.

La programación genética puede verse como cualquier método de evolución directa de programas de computadoras con el propósito de aprendizaje inductivo. Esta definición general hace que la programación genética sea independiente del modelo formal de cómputo utilizado para la representación de los programas evolucionados. En este tipo de programación, la programación genética, es frecuente el uso de un modelo de cómputo en particular, el árbol de cómputos, para representar los programas de computadoras. Sin embargo, en (Alonso et al., 2008), experimentan con otro modelo de cómputo, los programas de evaluación, y los resultados obtenidos, a decir de los autores, son alentadores y sugieren que el enfoque de la programación genética basado en programas de evaluación es considerablemente mejor que la programación genética convencional cuya estructura de representación está basada en árboles de cómputos. En (Alonso et al., 2008), además presentan sus resultados experimentales obtenidos al aplicar el enfoque de programación genética lineal, basado en programas de evaluación, en problemas de regresión simbólica y comparan con los resultados obtenidos en los mismos problemas mediante enfoques similares que utilizan codificación de árbol de cómputo.

Los programas de evaluación tienen aplicaciones en cuestiones tan diversas y amplias, como por ejemplo en la Geometría Algebraica, en la cual encontramos varias aplicaciones, una de ellas está dada en (Giusti et al., 1998), donde utilizan los programas de evaluación en la teoría de eliminación geométrica y presentan un método para resolver simbólicamente un sistema de ecuaciones polinomiales cero-dimensionales en el caso afín y tórico. En (Bank et al., 1996), presentan un trabajo basado en el trabajo de Giusti et al., cuyo objetivo es de mostrar como el método propuesto justamente en (Giusti et al., 1998), se puede aplicar a un caso de resolución de ecuaciones polinomiales reales.

En los últimos años se han realizado trabajos enfocados en la búsqueda de matrices MDS (del inglés *maximum distance separable*) implementables de manera eficiente para primitivas simétricas ligeras (del inglés *lightweight symmetric primitives*). Los últimos trabajos en este sentido, se centraron en optimizar localmente la multiplicación con elementos de una sola matriz. Aparte de esta línea de investigación, en (Kranz et al., 2017), se desarrollaron varias heurísticas para encontrar los programas de evaluación más cortos.

Los divisores de cantidades universales, lo que hoy llamamos el cálculo de factores de polinomios, ya fue pensado por Isaac Newton en 1673 y el método posteriormente publicado en su "Arithmetica Universali". En 1882, Leopold Kronecke, reduce el problema de factorizar polinomios de varias variables sobre campos algebraicos numéricos a factorizar polinomios de una sola variable sobre los números enteros, para lo cual aplicó el algoritmo de Newton. Van der

Waerden, en 1953, analiza esos algoritmos y sugiere que para problemas "más grandes" no son muy prácticos. No obstante, con los primeros programas de computadora que aplicaron este enfoque clásico, comprobaron que era bastante ineficiente. Los programas de evaluación como un medio para factorizar ciertos polinomios, se comenzó a desarrollar, en el marco de la Teoría de la Complejidad, desde la década de los años 70 aproximadamente, como se puede apreciar por ejemplo en (Paterson & Stockmeyer, 1973), (Strassen, 1974), (Heintz, 1986) y en varios otros artículos de la época. En (von zur Gathen, 1985), von zur Gathen combinó su teorema probabilístico de irreductibilidad de Hilbert con el método probabilístico de evaluación de programas de evaluación, (Ibarra & Moran, 1983), para encontrar un patrón en el grado de los factores de polinomios definidos mediante programas de evaluación. El problema de encontrar el máximo común divisor de polinomios, representados mediante programas de evaluación, es de tiempo polinomial probabilístico, como lo demostró Kaltofen en (Kaltofen, 1988), además en (Kaltofen, 1989), utilizó programas de evaluación para desarrollar algoritmos que permiten la fatorización de un polinomio de varias variables o multivariado en sus factores irreducibles y demuestra que los programas de evaluación para los factores irreducibles de un polinomio, dado por un programa de evaluación, también se pueden encontrar en tiempo polinomial probabilístico.

En (Schleimer, 2008), Saul Schleimer utiliza los programas de evaluación para demostrar que el problema de las palabras para los automorfismos de grupo de cualquier grupo libre finitamente generado es resoluble en tiempo polinomial. Esto nos demuestra que un programa de evaluación es una herramienta muy poderosa.

Uno de los trabajos relacionados con programas de evaluación sobre grupos más influyentes para este trabajo, se da en (Babai & Szemerédi, 1984). En dicho artículo, Babai y Szemerédi demostraron que en cualquier grupo G de orden n, cada elemento del grupo es generado mediante un programa de evaluación de longitud a lo sumo $(1 + \log n)^2$, es decir, de longitud $\mathcal{O}(\log^2 n)$. Cuando decimos que un elemento es generado por un programa de evaluación, nos referimos a que es computado por el programa de evaluación, de manera informal esto significa que si Γ_G denota un programa de evaluación en el grupo G cuyo conjunto generador es el conjunto S, y g es el elemento a computar, entonces existe un elemento $a \in S^n$ talque $\Gamma_G(S^n, a) = g$, es decir, existe una entrada $a = (a_1, \ldots, a_n)$, con $a_i \in S$ para $i = 1, \ldots, n$, tal que la salida de Γ_G para esta entrada es, justamente g. Más adelante, en la Sección 3.2, definiremos formalmente este concepto de computar un elemento de un grupo mediante un programa de evaluación.

CAPÍTULO 3

PROGRAMAS DE EVALUACIÓN

En este capítulo introducimos nuestro modelo formal de cómputo llamado programa de evaluación con entradas. En esta definición utilizamos la forma más general de programas de evaluación definida en (Bürgisser et al., 2013, Sección 4.1) especializada para grupos.

Antes de empezar con nuestras definiciones, recordemos previamente algunos conceptos básicos pero elementales de la Teoría de grupos.

3.1. Nociones de grupos

Un grupo es un conjunto G con una operación binaria $\cdot : G \times G \to G$ que cumple con las siguientes condiciones (Artin, 1991) y (Dorronsoro & Hernández, 1996):

1. *Asociatividad*: Para cualesquiera $a, b, c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

2. *Existencia del elemento neutro o identidad*: Existe $e \in G$ tal que $a \cdot e = e \cdot a = e$ para todo $a \in G$. Normalmente se utiliza 1 en vez de e.

3. *Existencia del elemento inverso*: Para cada $a \in G$, existe $b \in G$ tal que $a \cdot b = b \cdot a = e$. Por lo general se escribe a^{-1} en vez de b.

Se acostumbra a denotar $(G, \cdot)$ para referirse al grupo G en caso de que sea necesario especificar la operación de grupo $\cdot$, caso contrario, solo se escribirá G. El grupo G se dice *conmutativo* o *abeliano* si para cada $a, b \in G$, se cumple que $a \cdot b = b \cdot a$.

Notemos que al ser $\cdot$ una operación binaria en G, significa que $\cdot$ es una operación cerrada en G, es decir, si $a, b \in G$ tenemos que $a \cdot b \in G$. Si G cumple solo 1 y 2, decimos que G es un *monoide*.

Un subconjunto no vacío $S \subseteq G$ decimos que es un conjunto generador de G y lo denotamos por $G = \langle S \rangle$, si todo elemento $g \in G$ se puede escribir de la siguiente forma:

$$g = s_1^{\alpha_1} \cdot s_2^{\alpha_2} \cdots s_n^{\alpha_n}$$

donde $s_i \in S$, $\alpha_i \in \mathbb{Z}$ para cada $i = 1, \ldots, n$.

Veamos dos ejemplos de grupos que nos serán útiles más adelante.

Sean $(G, \cdot)$ y $(H, *)$ dos grupos y $f : G \to H$ una función. Decimos que f es un

1. *Homomorfismo* si $f(a \cdot b) = f(a) * f(b)$, para cada $a, b \in G$.
2. *Monomorfismo* si f es un homomorfismo inyectivo.
3. *Epimorfismo* si f es un homomorfismo sobreyectivo.
4. *Isomorfismo* si f es un homomorfismo biyectivo.
5. *Endomorfismo* si f es un homomorfismo y $G = H$.
6. *Automorfismo* si f es un isomorfismo y $G = H$.

Sea G un grupo y $H \subseteq G$ no vacío. Decimos que H es un subgrupo de G si se cumple que

1. $a \cdot b \in H$, para cada $a, b \in H$.
2. $a^{-1} \in H$ para cada $a \in H$.

Para indicar que H es un subgrupo de G, escribiremos $H \leqslant G$. Si $H, K \leqslant G$, entonces $H \cap K \leqslant G$. Si $f : G_1 \to G_2$ es un homomorfismo de grupos, $H_1 \leqslant G_1$ y $H_2 \leqslant G_2$, entonces $f(H_1) \leqslant G_2$ y $f^{-1}(H_2) \leqslant G_1$. Si f es un isomorfismo, entonces G_1 es conmutativo si, y solo si, G_2 es conmutativo.

Pasamos ahora a la definición de programas de evaluación.

3.2. Definiciones básicas

Definición 3.1 (Programas de evaluación)**.** Sea S un conjunto generador de un grupo G y $a = (a_1, \ldots, a_n) \in S^n$ con $n \in \mathbb{Z}^+$. Definimos un *programa de evaluación* en G, sobre S^n, como un par $\Gamma_G = (\Gamma, G)$ donde $\Gamma = (\Gamma_1, \ldots \Gamma_t)$ es una secuencia finita de instrucciones y $t \in \mathbb{Z}^+$ es su longitud. Cada instrucción Γ_i tiene la forma:

(i) $\Gamma_i(a) = a_j$ para algún $j \in \{1, \ldots, n\}$, o

(ii) $\Gamma_i(a) = \Gamma_j^{-1}(a)$ para algún $j < i$ en $\{1, \ldots, t\}$, o

(iii) $\Gamma_i(a) = \Gamma_j(a) \cdot \Gamma_k(a)$ para algún $j, k < i$ en $\{1, \ldots, t\}$.

Notemos que $\Gamma_j^{-1}(a)$ se refiere a $\Big(\Gamma_j(a)\Big)^{-1}$, es decir, el elemento inverso de $\Gamma_j(a)$, o sea, la inversa de una instrucción anterior.

Denotaremos por (S^n, a) la entrada para un programa de evaluación en G sobre S^n. Decimos que Γ_G es *ejecutable* en la entrada (S^n, a) con salida $b = (b_1, \ldots, b_t) \in G^t$ si $b_i = \Gamma_i(a)$ para $1 \leq i \leq t$; de forma más breve escribimos $b = \Gamma_G(S^n, a)$. Un programa de evaluación Γ_G *computa* un conjunto $F \subseteq G$ en S^n si y solo si $\Gamma_G(S^n, a) = b$ para alguna entrada (S^n, a) y $F \subseteq \{b_1, \ldots, b_t\}$. Si Γ_G es ejecutable en la entrada (S^n, a) con $\Gamma_G(S^n, a) = b$, diremos que Γ_G computa b en S^n.

De aquí en adelante nos referiremos a un programa de evaluación en G sobre S^n simplemente como un programa de evaluación sin precisar el grupo ni el conjunto generador, a menos que sea necesario, en cuyo caso sí estaremos indicando como en la definición.

La principal diferencia con la definición original de (Bürgisser et al., 2013) es que la entrada (S^n, a) no forma parte del conjunto computado F; de manera a incluir a (S^n, a) en F, acorde a la Definición 3.1, es necesario utilizar instrucciones en nuestro programa de evaluación. Esto es de manera a evitar que programas de evaluación de longitud 0 computen de forma gratuita la entrada.

La Definición 3.1 también es más general que la definición típica utilizada en la literatura del álgebra computacional. Por ejemplo, en la definición de (Krick, 2002), solo interesa el valor de la última instrucción del programa, mientras que en el trabajo de (Babai & Szemerédi, 1984), se consideran los resultados intermedios de todas las instrucciones y se toman como entradas cualquier número de elementos del conjunto generador.

La principal motivación de la Definición 3.1, y es la razón por la que adoptamos la definición de (Bürgisser et al., 2013), es la de obtener un modelo formal de computación cuya complejidad pueda expresarse en función a la longitud de la entrada. De esta forma, el programa de evaluación depende de la longitud de la entrada y esto da lugar a lo que en complejidad computacional se conoce como *computación no uniforme* (Arora & Barak, 2009, Capítulo 6).

Veamos ahora algunos ejemplos sencillos de programas de evaluación.

Ejemplo 3.1. Consideremos el grupo diedral $D_6 = \langle r, s \mid r^6 = 1, s^2 = 1, srs^{-1} = r^{-1} \rangle$ de orden 12. Un programa de evaluación $\Gamma_{D_6} = (\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, \Gamma_5)$ de longitud 5 que computa al elemento sr^3, es el siguiente

$$\begin{aligned}
\Gamma_1(a) &:= s && \longrightarrow \quad b_1 = s \\
\Gamma_2(a) &:= r && \longrightarrow \quad b_2 = r \\
\Gamma_3(a) &:= \Gamma_2(a) \cdot \Gamma_2(a) && \longrightarrow \quad b_3 = r^2 \\
\Gamma_4(a) &:= \Gamma_3(a) \cdot \Gamma_2(a) && \longrightarrow \quad b_4 = r^3 \\
\Gamma_5(a) &:= \Gamma_1(a) \cdot \Gamma_4(a) && \longrightarrow \quad b_5 = sr^3
\end{aligned}$$

donde $a \in S^2$ y $S = \{r, s\}$.

Ejemplo 3.2. Ahora consideremos el grupo simétrico S_6 de orden 6!. Sean $\alpha = (1\,2\,3\,4\,5\,6)$ y $\beta = (1\,2)$ dos permutaciones tales que $S_6 = \langle \alpha, \beta \rangle$. Un programa de evaluación Λ_{S_6} que compute el elemento $(1\,2\,3)(4\,5\,6)$ está dado por el siguiente conjunto de instrucciones

$$\begin{aligned}
\Lambda_1(a) &:= \alpha && \longrightarrow \quad b_1 = \alpha = (1\,2\,3\,4\,5\,6)\\
\Lambda_2(a) &:= \beta && \longrightarrow \quad b_2 = \beta = (1\,2)\\
\Lambda_3(a) &:= \Lambda_1(a) \cdot \Lambda_1(a) && \longrightarrow \quad b_3 = \alpha^2 = (1\,3\,5)(2\,4\,6)\\
\Lambda_4(a) &:= \Lambda_3(a) \cdot \Lambda_2(a) && \longrightarrow \quad b_4 = \alpha^2\beta = (1\,3\,5\,2\,4\,6)\\
\Lambda_5(a) &:= \Lambda_4(a) \cdot \Lambda_1(a) && \longrightarrow \quad b_5 = \alpha^2\beta\alpha = (1\,4)(2\,5\,3\,6)\\
\Lambda_6(a) &:= \Lambda_5(a) \cdot \Lambda_2(a) && \longrightarrow \quad b_6 = \alpha^2\beta\alpha\beta = (1\,4\,2\,5\,3\,6)\\
\Lambda_7(a) &:= \Lambda_6(a) \cdot \Lambda_6(a) && \longrightarrow \quad b_7 = \alpha^2\beta\alpha\beta\alpha^2\beta\alpha\beta = (1\,2\,3)(4\,5\,6).
\end{aligned}$$

con $a \in S^2$ y $S = \{\alpha, \beta\}$.

3.3. Propiedades de los programas de evaluación

Los programas de evaluación introducidos en la Definición 3.1 tienen varias propiedades interesantes. Algunas de estas propiedades las presentamos formalmente a continuación, algunas como proposiciones, lemas, corolarios y otras simplemente como comentarios debido a que tiende a lo trivial, pero su implicancia por más básica que sea ayuda a la comprensión de los programas de evaluación y el cómputo con este modelo.

Si un programa de evaluación Γ_G computa un conjunto generador de G para algún entero positivo n, entonces G es finitamente generado. Esto es debido a que la longitud de cualquier programa de evaluación es siempre finita.

Lema 3.1. Para cualquier grupo G y cualquier conjunto generador S de G, si $g \in G$, entonces existe un programa de evaluación Γ_G que computa g en S^n para algún entero positivo n.

Demostración. Sea S cualquier conjunto generador del grupo G. Si $g \in S$, entonces el programa de evaluación $\Gamma_1(a) = g$ computa g. Si $g \notin S$ y como $\langle S \rangle = G$, entonces podemos escribir

$$g = s_1^{\alpha_1} \cdot s_2^{\alpha_2} \cdots s_n^{\alpha_n} \tag{3.1}$$

para algún entero positivo n, con $s_i \in S$ y $\alpha_i \in \mathbb{Z}$ para $i = 1, \ldots, n$. A partir de aquí podemos construir trivialmente un programa de evaluación que compute g simplemente añadiendo una instrucción Γ_i para cada elemento s_i que aparece en la descripción (3.1) de g. □

Como un simple corolario del Lema 3.1 podemos decir que para cualquier subconjunto finito $F \subseteq G$ existe un programa de evaluación en G sobre S^n que computa F.

La siguiente proposición afirma la existencia de programas de evaluación óptimos, es decir, del conjunto de programas de evaluación que computan un conjunto dado, existe uno de longitud mínima, la menor posible.

Proposición 3.1. Para cualquier grupo G con conjunto generador S, si $g \in G$, entonces existe un programa de evaluación de longitud mínima que computa g en S^n para algún n.

Demostración. Si $g \in S$, entonces el programa de evaluación con una sola instrucción $\Gamma_1(a) := g$ computa g y es de longitud la menor posible. Si $g \notin S$, consideremos la ecuación (3.1) y supongamos, por el absurdo, que no existe un programa de evaluación de longitud mínima que computa g. Por lo tanto, para cualquier programa de evaluación Γ que computa g y de longitud t, existe otro programa de evaluación Λ que también computa g y cuya longitud s sea menor a t. Esto implica la existencia de un programa de evaluación de longitud 1 que computa g. De acuerdo a la Definición 3.1, el único programa de evaluación de longitud 1 que computa g es un programa de evaluación con una sola instrucción $\Gamma_1(a) := g$ con $g \in S$, lo cual es una contradicción ya que $g \notin S$. Luego, concluimos que existe un programa de evaluación de longitud mínima que computa g en S^n para algún entero positivo n. □

Proposición 3.2. Sean Γ_G y Λ_G dos programas de evaluación en G que computan conjuntos F_1 y F_2 en S_1^n y S_2^n, respectivamente. Existe un programa de evaluación Φ_G que computa $F_1 \cup F_2$ en S_3^{2n} y su longitud es la suma de las longitudes de Γ_G y Λ_G.

Demostración. Sean $\Gamma_G = (\Gamma_1, \ldots, \Gamma_r)$ y $\Lambda_G = (\Lambda_1, \ldots, \Lambda_s)$. Supongamos que Γ_G computa F_1 en S_1^n con entrada (S_1^n, a) y Λ_G computa F_2 en S_2^n con entrada (S_2^n, b). Definimos el programa de evaluación $\Phi_G = (\Gamma_1, \ldots, \Gamma_r, \Lambda_1, \ldots, \Lambda_s)$ de longitud $r + s$. Entonces Φ_G computa $F_1 \cup F_2$ en la entrada (S_3^{2n}, ab) donde ab es la concatenación de a y b y $S_3 = S_1 \cup S_2$ es un conjunto generador de G. □

Corolario 3.1. El programa Φ_G de la Proposición 3.2 computa $F_1 \cap F_2$ en S_3^{2n}.

El Corolario 3.1 es fácil de ver que es cierto porque la intersección de F_1 y F_2 está contenida en su unión.

3.4. Complejidad de evaluación de conjuntos

La motivación de la Definición 3.1 y la razón por la que nos basamos en (Bürgisser et al., 2013) es que queríamos un modelo computacional cuya longitud pueda depender de la longitud o tamaño de la entrada. De aquí en adelante estudiamos la complejidad computacional de computar conjuntos utilizando la longitud o tamaño como nuestro recurso computacional. A continuación definimos el costo de evaluación de un subconjunto F de un grupo G mediante un programa de evaluación Γ_G.

Definición 3.2 (Costo de evaluación)**.** El *costo de evaluación* de un conjunto F, denotado por $L_G(F, S^n)$, es la longitud mínima de un programa de evaluación Γ_G que computa F en S^n. Si $F \nsubseteq G$ entonces $L_G(F, S^n) = \infty$.

De acuerdo con la Definición 3.2, es claro que $L_G(F, S^n) \geq |F|$ para cualquier subconjunto $F \subseteq G$ y además según la Proposición 3.1, existe un programa de evaluación de longitud mínima que computa F en S^n.

En el trabajo seminal de (Babai & Szemerédi, 1984) se demostró que para cualquier grupo de orden n, cada elemento puede computarse mediante un programa de evaluación cuya longitud es a lo sumo $(1 + \log n)^2$.

Ahora definimos en base al costo de evaluación, la medida que estaremos utilizando en este trabajo para realizar el análisis de comparación entre el computo conmutativo y no conmutativo mediante un programa de evaluación. Esta medida estará definida sobre una familia de grupos, en donde cada grupo deberá contener al conjunto F, de esta manera se puede hablar de los programas de evaluación sobre familias de grupos que computan a dicho conjunto F.

Definición 3.3 (Complejidad de evaluación)**.** Dado un conjunto F, sea $\mathcal{G}$ una familia de grupos tal que $F \subseteq \bigcap_{G \in \mathcal{G}} G$. Para cada entero positivo n definimos la *complejidad de evaluación* de F con respecto a la familia $\mathcal{G}$ como:

$$\mathcal{L}_{\mathcal{G}}(F, n) = \min_{G \in \mathcal{G}} L_G(F, S^n).$$

donde S es un conjunto generador minimal de G.

En particular, si $\mathcal{C}$ es la familia de todos los grupos conmutativos que contienen a F y $\mathcal{N}$ es la familia de todos los grupos no conmutativos que contienen a F, definimos:

$$\mathcal{L}_c(F, n) = \min_{G \in \mathcal{C}} L_G(F, S^n) \quad \text{y} \quad \mathcal{L}_{nc}(F, n) = \min_{G \in \mathcal{N}} L_G(F, S^n).$$

En este trabajo llamamos a $\mathcal{L}_c$ y $\mathcal{L}_{nc}$ la complejidad de evaluación conmutativa y la complejidad de evaluación no conmutativa de F, respectivamente. Si $\mathcal{G}$ es la familia de todos los grupos que contienen a F, definimos análogamente la complejidad total $\mathcal{L}$ de F donde el mínimo se toma sobre todos los grupos definibles que le contengan a G.

El siguiente lema nos muestra una relación muy simple pero a la vez importante para dos familias de grupos.

Lema 3.2. Sean $\mathcal{G}'$ y $\mathcal{G}$ dos familias de grupos tales que F es un subconjunto de cada miembro de $\mathcal{G}'$ y $\mathcal{G}$. Si $\mathcal{G}' \subseteq \mathcal{G}$, entonces $\mathcal{L}_{\mathcal{G}}(F, n) \leq \mathcal{L}_{\mathcal{G}'}(F, n)$.

La demostración del lema anterior es inmediata, debido a que si $G' \in \mathcal{G}'$ y $\Gamma_{G'}$ es un programa de evaluación y como $\mathcal{G}' \subseteq \mathcal{G}$, entonces $G' \in \mathcal{G}$ y $\mathcal{L}_{\mathcal{G}}(F, n) \leq \mathcal{L}_{\mathcal{G}'}(F, n)$.

Proposición 3.3. Para cualquier par de conjuntos F y F' y cualquier grupo G que los contenga, se verifica que

$$L_G(F \cup F', (S \cup T)^{n+m}) \leq L_G(F, S^n) + L_G(F', T^m),$$

donde $\langle S \rangle = \langle T \rangle = G$ y $n,\ m$ enteros positivos.

Demostración. Sean $\Gamma = (\Gamma_1, \ldots, \Gamma_r)$ y $\Lambda = (\Lambda_1, \ldots, \Lambda_s)$ dos programas de evaluación que computan los conjuntos F y F' respectivamente. Supongamos además que Γ computa F en S^n con entrada (S^n, a) y Λ computa F' en T^m con entrada (T^m, b). Ahora definimos un nuevo programa de evaluación $\Phi = (\Gamma_1, \ldots, \Gamma_r, \Lambda_1, \ldots, \Lambda_s)$ de longitud $r + s$. Entonces Φ computa $F \cup F'$ con entrada $\big((S \cup T)^{m+n}, ab\big)$ donde ab es la concatenación de a y b y $\langle S \cup T \rangle = G$. □

Los siguientes dos corolarios son consecuencias inmediatas de la Proposición 3.3.

Corolario 3.2. Para cualquier familia $\mathcal{G}$ de grupos tal que $F, F' \subseteq \bigcap_{G \in \mathcal{G}} G$ se verifica que

$$\mathcal{L}_{\mathcal{G}}(F \cup F', n + m) \leq \mathcal{L}_{\mathcal{G}}(F, n) + \mathcal{L}_{\mathcal{G}}(F', m).$$

Corolario 3.3. $L_G(F \cap F', (S \cup T)^{n+m}) \leq L_G(F, S^n) + L_G(F', T^m)$.

3.5. Simulación de programas de evaluación

En esta sección presentamos una técnica que llamaremos simulación de programas de evaluación que nos va a permitir relacionar la complejidad de evaluación entre diferentes familias de grupos. El término "simulación" está inspirada en el comportamiento de una máquina de Turing universal, donde una máquina simula el comportamiento de otra máquina.

Para introducir el concepto de simulación primero necesitamos definir la idea de extensión isomorfa de un subonjunto de un grupo a otro grupo, el cual nos va a permitir relacionar programas de evaluación sobre diferentes grupos. En adelante escribiremos $H \leqslant G$ para indicar que H es un subgrupo de G.

Definición 3.4 (Extensión isomorfa)**.** Sean G y G' dos grupos. Decimos que un subconjunto $F \subseteq G$ tiene una extensión F' en G' si y solo si existe un subgrupo $H \leqslant G$ que contiene a F, un subgrupo $H' \leqslant G'$ que contiene a F', y un homomorfismo sobreyectivo $\phi : H \to H'$ tal que $\phi(F) = F'$. Llamamos a F' la *extensión homomorfa* (o simplemente *extensión*) de F en G' y lo denotaremos como $\hat{F}_{G'}$ o simplemente $\hat{F}$ cuando el grupo queda claro. Si ϕ es un isomorfismo, entonces diremos que $\hat{F}_{G'} = F'$ es una *extensión isomorfa* de F en G'.

Ahora podemos definir apropiadamente nuestra idea de simulación de programas de evaluación.

Definición 3.5 (Simulación)**.** Sea $G = \langle S \rangle$ y $G' = \langle R \rangle$ dos grupos y sea Γ_G un programa de evaluación. Para cualquier entero positivo n, decimos que un programa de evaluación $\Lambda_{G'}$ *simula* a Γ_G si y solo si para todo conjunto F computable por Γ_G en S^n, existe una extensión isomorfa $\hat{F}_{G'}$ de F en G' que es computable por $\Lambda_{G'}$ en $R^{f(n)}$, donde $f : \mathbb{N} \to \mathbb{N}$ y $f(n) \geq n$. Esto lo denotamos escribiendo $\Gamma_G \prec \Lambda_{G'}$.

Si dos grupos no contienen subgrupos isomorfos, entonces de la Definición 3.5 de simulación es fácil ver que no puede existir una simulación entre ellos. Es decir, si los grupos G y G' no

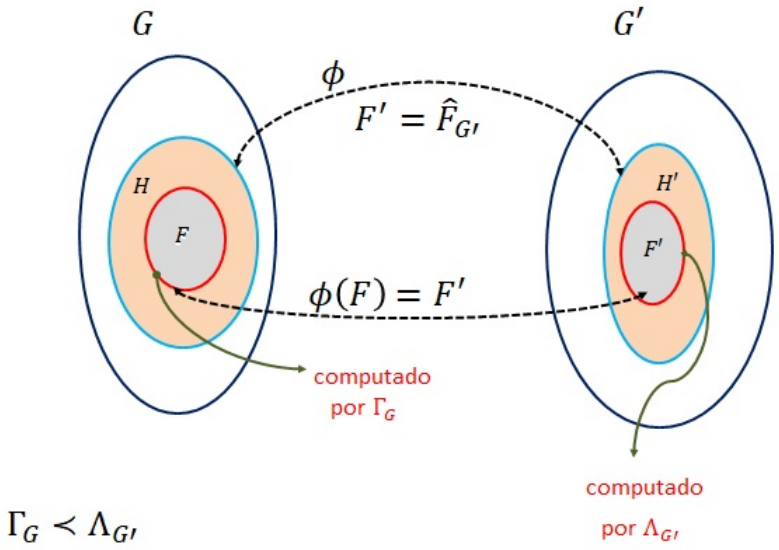

Figura 3.1: Simulación

contienen subgrupos isomorfos, entonces no existen programas de evaluación Γ_G y $\Lambda_{G'}$ tal que $\Gamma_G \prec \Lambda_{G'}$.

Como la simulación establece una relación entre programas de evaluación, es natural preguntarnos que tipo de relación es. Los siguientes resultados nos brindarán una visión clara al respecto.

Lema 3.3. La relación $\prec$ es una relación reflexiva. Esto es, $\Gamma_G \prec \Gamma_G$.

La demostración es trivial, ya que si F es un conjunto computable por Γ_G en S^n, F es un subconjunto de G y G es isomorfo a si mismo, entonces F es una extensión isomorfa de F en G.

Lema 3.4. La relación $\prec$ es una relación transitiva. Esto es, si $\Gamma_{G_1} \prec \Lambda_{G_2}$ y $\Lambda_{G_2} \prec \Phi_{G_3}$, entonces $\Gamma_{G_1} \prec \Phi_{G_3}$.

Demostración. Sea F_1 un conjunto computado por Γ_{G_1} en S_1^n y $\hat{F}_{1G_2}$ una extensión isomorfa de F_1 en G_2 que es computable por Λ_{G_2} en S_2^n. De la misma manera, sea $\hat{F}_{2G_3}$ una extensión isomorfa de F_2 en G_3 computable por Φ_{G_3} en S_3^n donde $G_1 = \langle S_1 \rangle$, $G_2 = \langle S_2 \rangle$ y $G_3 = \langle S_3 \rangle$. Para demostrar la transitividad de $\prec$ es suficiente con demostrar que F_3 es una extensión isomorfa de F_1 en G_3.

Sean $H_1 \leqslant G_1$, $H_2 \leqslant G_2$ y $\phi : H_1 \to H_2$ un isomorfismo tal que $F_1 \subseteq H_1$, $F_2 \subseteq H_2$ y $\phi(F_1) = F_2$. Análogamente, sean $N_2 \leqslant G_2$, $N_3 \leqslant G_3$ y $\varphi : N_2 \to N_3$ un isomorfismo tal que $F_2 \subseteq N_2$, $F_3 \subseteq N_3$ y $\varphi(F_2) = F_3$. Como H_2 y N_2 son subgrupos de G_2, entonces $H_2 \cap N_2$ es también un subgrupo de G_2. Notemos que $F_2 \subseteq H_2 \cap N_2$ y que $\phi^{-1}(H_2 \cap N_2)$ es un subgrupo de G_1 y $\varphi(H_2 \cap N_2)$ es un subgrupo de G_3. Luego tenemos que $F_1 = \phi^{-1}(F_2)$ y $\varphi(F_2) = F_3$. Esto implica que $\varphi \circ \phi$ es un isomorfismo entre $\phi^{-1}(H_2 \cap N_2)$ y $\varphi(H_2 \cap N_2)$ y se cumple que $\varphi(\phi(F_1)) = F_3$. Concluimos que F_3 es una extensión isomorfa de F_1 en G_3, como queríamos. □

A continuación, con la Proposición 3.4 determinamos que la relación de simulación es una relación de preorden.

Proposición 3.4. La relación $\prec$ es una relación de preorden.

La Proposición 3.4 de arriba es consecuencia inmediata de los Lemas 3.3 y 3.4 anteriores.

La siguiente proposición nos permite concluir que la relación de simulación no es una relación de equivalencia.

Proposición 3.5. La relación $\prec$ no es una relación simétrica.

Demostración. Sea G cualquier grupo no trivial y sea $\{1\}$ el grupo trivial. Cualquier programa de evaluación $\Lambda_{\{1\}}$ puede ser simulado por cualquier programa de evaluación Γ_G, y por lo tanto, $\Lambda_{\{1\}} \prec \Gamma_G$. Por otro lado, si F es un conjunto computado por Γ_G que contenga por lo menos un elemento que no sea la identidad, entonces ningún programa de evaluación en $\{1\}$ podrá computar F. □

3.6. Complejidad conmutativa versus complejidad no conmutativa

La medida definida en la Definición 3.3, como hemos mencionado antes, es la medida que estaremos utilizando para comparar el poder de cómputo de los programas de evaluación conmutativos y no conmutativos. En esta sección estudiaremos la relación entre la complejidad de evaluación conmutativa y la no conmutativa.

Lema 3.5. Para todo grupo conmutativo G con $F \subset G$ existe un grupo no conmutativo G' tal que

$$L_{G'}(\hat{F}_{G'}, R^n) \leq L_G(F, S^n)$$

donde $\langle S \rangle = G$ y $\langle R \rangle = G'$.

Demostración. Sea Γ_G un programa de evaluación de longitud mínima que computa F en la entrada $a = (a_1, \ldots, a_n) \in S^n$ para un conjunto generador S de G y con salida $b = (b_1, \ldots, b_t) \in G^t$. Construimos un grupo no conmutativo G' y un programa de evaluación $\Lambda_{G'}$ que simula Γ_G.

Sea N cualquier grupo no conmutativo con 1_N como elemento identidad y sea $G' = G \times N$. Es claro que G' es un grupo con la operación de grupo definida como una multiplicación componente a componente, y además que G' es no conmutativo pues N no lo es. Sea ahora $\Gamma_G = (\Gamma_1, \ldots, \Gamma_t)$. Para cada instrucción Γ_i construimos una nueva instrucción Λ_i en G' de la siguiente manera:

(i) si $\Gamma_i(a) = a_j$ para algún $j \in \{1, \ldots, n\}$, entonces hacemos $\Lambda_i = (a_j, 1_N)$;

(ii) Si $\Gamma_i(a) = \Gamma_j^{-1}(a)$ para algún $j < i$, entonces hacemos $\Lambda_i = (\Gamma_j^{-1}, 1_N)$;

(iii) si $\Gamma_i(a) = \Gamma_j(a) \cdot \Gamma_k(a)$ para algún $j, k < i$, entonces hacemos $\Lambda_i = (\Gamma_j \cdot \Gamma_k, 1_N)$.

Sea $F = \{b_1, \ldots, b_t\}$. Si definimos la entrada para $\Lambda_{G'}$ como $a' = \big((a_1, 1_N), \ldots, (a_n, 1_N)\big)$, entonces la salida es exactamente $b' = \big((b_1, 1_N), \ldots, (b_t, 1_N)\big)$.

Consideremos ahora $F' = \{(b_1, 1_N), \ldots, (b_t, 1_N)\}$. Sea $G_1 = G \times \{1_N\}$. Tenemos entonces que G_1 es un subgrupo de G' y es isomorfo a G. Entonces como $F' \subseteq G_1$ y $F \subseteq G$, tenemos que F' es una extensión isomorfa de F en G', es decir, $F' = \hat{F}_{G'}$. Concluimos que $\Gamma_G \prec \Lambda_{G'}$ y el número de instrucciones de $\Lambda_{G'}$ es igual al número de instrucciones de Γ_G. Por lo tanto, como Γ_G es de longitud mínima, tenemos que $L_{G'}(\hat{F}_{G'}, R^n) \leq L_G(F, S^n)$. □

Uno de los resultados principales de este trabajo lo presentamos en el siguiente teorema.

Teorema 3.1. $\mathcal{L}_{nc}(\hat{F}, n) \leq \mathcal{L}_c(F, n)$.

Demostración. La demostración del teorema es una consecuencia inmediata del Lema 3.5. Pues notemos que según el lema mencionado, para todo grupo conmutativo G que contiene a F existe un grupo no conmutativo G' que contiene una extensión isomorfa $\hat{F}_{G'}$ de F en G' tal que

$$L_{G'}(\hat{F}_{G'}, R^n) \leq L_G(F, S^n),$$

entonces podemos afirmar que

$$\min_{G' \in \mathcal{N}} L_{G'}(\hat{F}_{G'}, R^n) \leq \min_{G \in \mathcal{C}} L_G(F, S^n)$$

es decir

$$\mathcal{L}_{nc}(\hat{F}, n) \leq \mathcal{L}_c(F, n)$$

como queríamos. □

El Teorema 3.1 establece que la complejidad de evaluación no conmutativa de un conjunto F es a lo sumo la complejidad de evaluación conmutativa del conjunto F.

CAPÍTULO 4

PROGRAMAS DE EVALUACIÓN EN GRUPOS TOPOLÓGICOS

En este capítulo consideraremos un programa de evaluación definido en un grupo topológico y presentaremos algunos resultados interesantes que se obtienen bajo esa consideración. Primeramente recordaremos algunas definiciones topológicas básicas, luego veremos algunos conceptos básicos necesarios de grupos topológicos y algunas propiedades importantes de estos. Inmediatamente después definiremos los programas de evaluación sobre grupos topológicos y pasamos a analizar algunas propiedades intrínsecas relacionadas a estos en dichos espacios. Estudiamos al final algunas propiedades interesantes que resultan al considerar los programas de evaluación en grupos topológicos.

4.1. Espacios topológicos

Sea X un conjunto no vacío, una topología (o estructura topológica) en X es una familia o colección τ de subconjuntos de X que satisface las siguientes tres propiedades:

1. $\emptyset$ y X están en τ.

2. La intersección finita de elementos de τ está en τ.

3. La unión arbitraria de elementos de τ está en τ.

Al par (X, τ) consistente de un conjunto X y una topología τ en X (sobre X o de X, indistintamente) se denomina *espacio topológico*. Cuando no es necesario especificar la topología con la cual el conjunto X se considera como un espacio topológico, simplemente nos referiremos a X como un espacio, sin confundirlo con un simple conjunto. Notemos que dado un conjunto

X, este puede estar munido o dotado de varias topologías distintas entre sí y que en cada caso lo convierte en un espacio topológico distinto.

A los elementos del espacio X se los llama *puntos* y a cada miembro o elemento de la topología τ se lo denomina *conjunto abierto* o simplemente un *abierto*, esto es, U es abierto en X si y sólo si, $U \in \tau$. De acuerdo a esto y a lo mencionado en el párrafo anterior, un subconjunto puede ser abierto para una topología y no ser abierto para otra topología.

Si (X, τ) es un espacio topológico, un entorno de un punto $x \in X$ es cualquier conjunto abierto que contenga a x, según se considera en (Dugundji, 1966) y (Munkres, 2000).

Cualquier conjunto X puede dotarse de una topología, por ejemplo, si consideramos $\tau_1 = \mathscr{P}(X)$ la colección de todos los subconjuntos de X y $\tau_2 = \{\emptyset, X\}$, ambas son topologías sobre X y son llamadas topología discreta y topología indiscreta o trivial, respectivamente. Así, (X, τ_1) se denomina espacio discreto y (X, τ_2) espacio indiscreto.

Sea (X, τ) un espacio topológico. Cualquier subconjunto $A \subseteq X$ puede dotarse de una topología que hereda como subconjunto de X. En efecto, la familia

$$\tau_A := \{A \cap U \mid U \in \tau\}$$

es una topología sobre el conjunto A denominada *topología de subespacio*, *topología relativa* o *topología inducida* de A respecto a X. Con esta topología, A se denomina subespacio topológico de X; sus conjuntos abiertos son todas las intersecciones de conjuntos abiertos de X con A. Este hecho nos muestra con claridad que el ser un conjunto abierto es relativo a la topología en cuestión, pues si $A \subseteq Y \subseteq X$, puede darse el caso que A sea abierto en Y pero no así en X. Ahora, si A es abierto en X, de hecho lo es también en Y, pues $A = Y \cap A$ con A abierto en X.

La tarea de especificar una topología se simplifica bastante dando solo los conjuntos abiertos necesarios para generar todos los demás conjuntos abiertos. Esto nos introduce a la definición de base para una topología. Sea (X, τ) un espacio topológico. Una familia $\mathscr{B} \subseteq \tau$ se denomina una *base* para τ si cada conjunto abierto es la unión de miembros de $\mathscr{B}$. $\mathscr{B}$ es también llamado una base para el espacio X y cada miembro de la base se denomina *conjunto abierto básico* de la topología τ. No solo es cada miembro de τ la unión de miembros de $\mathscr{B}$, sino también, debido a que $\mathscr{B} \subseteq \tau$ y a la definición de topología, cada unión de elementos de $\mathscr{B}$ pertenece a τ, esto significa que una base para τ determina completamente a τ.

Dado una base $\mathscr{B}$ para la topología de X, puede existir varias bases para la misma topología en X. A partir de una base $\mathscr{B}$ podemos generar u obtener también la topología τ en X que tiene como base a $\mathscr{B}$ de la siguiente manera:

$$\tau := \{U \subseteq X \mid \forall x \in U,\ \exists\, \beta \in \mathscr{B} :\ x \in \beta \subseteq U\}$$

Dos bases $\mathscr{B}$ y $\mathscr{B}'$ en X son equivalentes si ambas generan la misma topología sobre X.

Una subbase $\mathscr{S}$ para una topología sobre X es una familia o colección de subconjuntos de X cuya unión es igual a X. La topología generada por la subbase $\mathscr{S}$ se define como la colección de todas las uniones de intersecciones finitas de los elementos de $\mathscr{S}$. De acuerdo a la definición

de subbase, podemos concluir que si $\mathscr{S}$ es una subbase, entonces $\mathscr{S} \subseteq \mathscr{B} \subseteq \tau$ donde

$$\begin{aligned} \mathscr{B} &:= \{ \text{ intersecciones finitas de elementos de } \mathscr{S} \} \\ \tau &:= \{ \text{ uniones arbitrarias de elementos de } \mathscr{B} \} \end{aligned}$$

Dados dos espacios topológicos (X, τ_X) y (Y, τ_Y), la topología producto sobre el producto cartesiano $X \times Y$ es la topología que tiene como base a la colección de todos los conjuntos de la forma $U \times V$ donde U es abierto en X y V es abierto en Y, es decir, si $U \in \tau_X$ y $V \in \tau_Y$. Esta definición se puede generalizar para cualquier producto cartesiano de n espacios topológicos $X_1, \ldots, X_n$, esto es, la topología producto del producto cartesiano finito

$$\prod_{i=1}^{n} X_i$$

es la topología que tiene como base a todos los conjuntos de la forma

$$\prod_{i=1}^{n} V_i$$

donde cada V_i es un conjunto abierto en X_i, para $i = 1, \ldots, n$.

Consideremos un espacio topológico X y recordemos algunos conceptos elementales y familiares en cualquier curso de análisis.

Un subconjunto $A \subseteq X$ se llama *conjunto cerrado* si el complemento de A en X, que lo vamos a denotar aquí mediante $\mathscr{C}_X(A) = X - A$, es un conjunto abierto en X.

Es importante notar que

(1) La intersección arbitraria de cualquier familia de conjuntos cerrados es un conjunto cerrado.

(2) La unión de una cantidad finita de conjuntos cerrados es un conjunto cerrado.

Esta definición de arriba de conjunto cerrado no es intrínseca, ya que para decidir si es cerrado o no un conjunto se considera el complemento y no el conjunto en sí. Veamos ahora algunas formulaciones intrínsecas mediante otros conceptos que nos serán útiles más adelante.

Sea $A \subseteq X$. Un punto $x \in X$ se denomina *punto de adherencia* de A si todo entorno de x interseca a A, es decir, $U \cap A \neq \emptyset$. Al conjunto de todos los puntos de adherencia de A se lo denota por $\overline{A}$ y se lo llama la clausura de A. De esta manera podemos escribir

$$\overline{A} := \{x \in X \mid \forall \text{ entorno } U \text{ de } x, \, : U \cap A \neq \emptyset\}$$

Otra forma de definir $\overline{A}$ es la siguiente:

$$\overline{A} := \bigcap \{F \subseteq X \mid (F \text{ es cerrado }) \wedge \ F \supseteq A\}$$

Mediante esta última igualdad, es fácil ver que

1. $A \subseteq \overline{A}$.
2. A es cerrado si, y solo si, $A = \overline{A}$.
3. $\overline{A}$ es el menor conjunto cerrado que contiene a A, es decir, si F es un conjunto cerrado que contiene a A, entonces $F \subseteq \overline{A}$.
4. Si $A \subseteq B$, entonces $\overline{A} \subseteq \overline{B}$.
5. $\overline{\overline{A}} = \overline{A}$, esto es, $\overline{A}$ es un conjunto cerrado.

Otra forma alternativa de describir los conjuntos cerrados es a través de sus puntos de acumulación.

Un punto $x \in X$ se denomina *punto de acumulación* o *punto límite* de un conjunto A si cada entorno U de x contiene al menos un punto de A distinto a x, es decir, si $U \cap (A - \{x\}) \neq \emptyset$. Al conjunto de todos los puntos de acumulación se lo denota por A' y se llama conjunto derivado de A. Así

$$A' := \{x \in X \mid \forall \text{ entorno } U \text{ de } x, : U \cap (A - \{x\}) \neq \emptyset\}$$

Las propiedades fundamentales del conjunto derivado son las siguientes:

1. $\overline{A} = A \cup A'$
2. A es cerrado si, y solo si, $A' \subseteq A$, es decir, A es cerrado si, y solo si, contiene a todos sus puntos de acumulación.

Con la idea de relacionar diferentes espacios topológicos introducimos el concepto bien conocido ya de función entre espacios topológicos. Dados dos espacios topológicos (X, τ_X) e (Y, τ_Y), observemos que una función $f : X \to Y$ relaciona los elementos del conjunto X con los elementos del conjunto Y y también relaciona los subconjuntos de X (elementos de $\mathscr{P}(X)$) con los subconjuntos de Y (elementos de $\mathscr{P}(Y)$) y viceversa, y esto lo hace ya que induce dos funciones $2^f : \mathscr{P}(X) \to \mathscr{P}(Y)$ y $2^{f^{-1}} : \mathscr{P}(Y) \to \mathscr{P}(X)$ tales que para cada $A \subseteq X$ y cada $B \subseteq Y$,

$$2^f(A) = f(A) \quad \text{y} \quad 2^{f^{-1}}(B) = f^{-1}(B),$$

es decir, 2^f es la función imagen y $2^{f^{-1}}$ es la función preimagen o imagen inversa. Notemos que $2^{f^{-1}}$ en absoluto se refiere a la función inversa de f sino a la preimagen de B mediante f. A partir de aquí forzamos un poco la notación pues, por simplicidad, para expresar $2^f(A)$ escribiremos $f(A)$ y para expresar $2^{f^{-1}}(B)$ escribiremos $f^{-1}(B)$.

En base a lo mencionado, definamos la continuidad de funciones. Sea (X, τ_X) y (Y, τ_Y) dos espacios topológicos. Una función $f : X \to Y$ decimos que es continua si la preimagen de cada conjunto abierto en Y es un conjunto abierto en X, esto es, para cada $V \in \tau_Y$, $f^{-1}(V) \in \tau_X$, es decir, $2^{f^{-1}}$ envía elementos de τ_Y a elementos de τ_X. De esta manera podemos decir que las funciones $f : X \to Y$ que son adecuadas o interesantes, son aquellas que relacionan los elementos de τ_Y con los elementos de τ_X.

Algunas propiedades elementales de funciones continuas son las siguientes:

1. Composición: Si $f : X \to Y$ y $g : Y \to Z$ son funciones continuas, también lo es la función compuesta $g \circ f : X \to Z$.

2. Restricción del dominio: Si $f : X \to Y$ es continua y $A \subseteq X$ con la topología de subespacio, entonces $f|_A : A \to Y$ también es continua. Recordemos que $f|_A(x) = f(x)$ si $x \in A$.

3. Restricción del codominio: Si $f : X \to Y$ es continua y $f(X) \subseteq Y$ está dotado de su topología de subespacio, entonces $f : X \to f(X)$ también es continua.

Dado el producto cartesiano $X \times Y$, podemos definir las funciones

$$p_1 : X \times Y \to X \quad \text{y} \quad p_2 : X \times Y \to Y$$

mediante las ecuaciones:

$$p_1(x, y) = x \quad \text{y} \quad p_2(x, y) = y$$

para cada $(x, y) \in X \times Y$. Estas funciones se denominan funciones proyecciones en la primera y segunda coordenada, respectivamente. Estas funciones las podemos generalizar para el caso del producto cartesiano finito de n espacios X_i, $\prod_{i=1}^{n} X_i$, de la siguiente manera:

$$p_j : \prod_{i=1}^{n} X_i \to X_j \ \text{ donde } \ p_j(x) = x_j \ \text{ con } \ x \in \prod_{i=1}^{n} X_i \ \text{ y } j = 1, \ldots, n.$$

Si U_j es un abierto en X_j, notemos que $p_j^{-1}(U_i) = \prod_{i=1}^{n} Y_i$ donde

$$Y_i = \begin{cases} X_i & \text{si } i \neq j \\ U_j & \text{si } i = j \end{cases}$$

es decir

$$\begin{aligned} p_1^{-1}(U_1) &= U_1 \times X_2 \times \cdots \times X_n \\ p_2^{-1}(U_2) &= X_1 \times U_2 \times X_3 \cdots \times X_n \\ &\vdots \\ p_n^{-1}(U_n) &= X_1 \times X_2 \times \cdots \times U_n. \end{aligned}$$

Esto significa que si U_j es un conjunto abierto en X_j, entonces $p_j^{-1}(U_j)$ es un conjunto abierto en $\prod_{i=1}^{n} X_i$, donde $\prod_{i=1}^{n} X_i$ tiene la topología producto. Por lo que concluimos que las funciones proyecciones p_j son funciones continuas cualquiera sea $j = 1, \ldots, n$ siempre y cuando $\prod_{i=1}^{n} X_i$ esté dotado de la topología producto. Este último hecho nos permite afirmar con toda

certeza que la colección

$$\mathscr{S} = \{p_j^{-1}(U_j) \mid U_j \text{ abierto en } X_j \text{ con } j = 1, \ldots, n\}$$

es una subbase para la topología producto sobre el producto cartesiano $\prod_{i=1}^{n} X_i$, pues

$$\bigcup_{j=1}^{n} p_j^{-1}(U_j) = \prod_{i=1}^{n} X_i$$

y además

$$\bigcap_{j=1}^{n} p_j^{-1}(U_j) = \prod_{j=1}^{n} U_j$$

el cual es un elemento de la base para la topología producto de $\prod_{i=1}^{n} X_i$.

Una función continua $f : X \to Y$ es llamada un *homeomorfismo* si existe una función continua $g : Y \to X$ tal que

$$f \circ g = \mathbb{1}_Y \text{ y } g \circ f = \mathbb{1}_X$$

donde $\mathbb{1}_X$ y $\mathbb{1}_Y$ denotan la función identidad en X e Y, respectivamente. En otras palabras, f es un homeomorfismo si f es biyectiva y bicontinua (esto último significa que f y f^{-1} son funciones continuas).

Un espacio topológico X se dice que es homogéneo si para cada par de puntos $x, y \in X$ existe un homeomorfismo $f : X \to X$ tal que $f(x) = y$, es decir, para cada $x, y \in X$ existe un homeomorfismo que lleva x a y, y viceversa.

Sea $A \subseteq X$. Dada una familia $\mathcal{U}$ de subconjuntos de un espacio topológico X, decimos que $\mathcal{U}$ es un cubrimiento de A en X si

$$A \subseteq \bigcup_{U \in \mathcal{U}} U.$$

Si los elementos de la familia $\mathcal{U}$ son conjuntos abiertos de X, entonces decimos que $\mathcal{U}$ es un cubrimiento abierto de A en X. Decimos que un subconjunto $A \subseteq X$ es compacto en X si todo cubrimiento abierto $\mathcal{U}$ de A en X posee una subfamilia finita que también cubre a A.

4.2. Grupos topológicos

Un grupo topológico es una terna $(G, \cdot, \tau)$ consistente de un grupo G, la operación de grupo $\cdot : G \times G \to G$ y una topología τ en G tal que se cumpla las siguientes tres condiciones: (McCarty, 1988)

1. La función $\cdot$ es continua.

2. La función de inversión $inv : G \to G$ definida por $inv(g) = g^{-1}$ para cada $g \in G$, es continua.

3. El conjunto unitario $\{e\}$ es cerrado en G, con e el elemento neutro de G. Este requerimiento consiste en que $G - \{e\}$ sea un conjunto abierto en G.

Nos referiremos al grupo topológico $(G, \cdot, \tau)$ simplemente como el grupo topológico G sin la necesidad de especificar la operación de grupo ni la topología τ, por simplicidad, a menos que sea estrictamente necesario.

Para la continuidad de la función $\cdot : G \times G \to G$, consideramos $G \times G$ con su topología producto.

Todo subgrupo de un grupo topológico es un grupo topológico con respecto a la topología relativa. Cualquier grupo G puede considerarse un grupo topológico con respecto a la topología discreta llamado grupo discreto, y con respecto a la topología indiscreta es un grupo topológico llamado grupo indiscreto.

Alguno ejemplos interesantes de grupos topológicos son los siguientes.

1. El grupo aditivo $(X, +)$ de todo espacio normado $(X, \|\cdot\|)$ es un grupo topológico.

2. $(\mathbb{R}^n, +)$ es un grupo topológico abeliano con respecto a cualquier métrica inducida por la norma.

3. $(\mathbb{C}^\times, \cdot)$ es un grupo topológico.

4. El grupo $GL_n(\mathbb{R})$ de la matrices invertibles $n \times n$ es un grupo topológico con respecto a la multiplicación de matrices llamado el grupo general lineal de orden n sobre $\mathbb{R}$. La topología en cuestión es la topología inducida por la métrica

$$d(A, B) = \left(\sum_{i,j=1}^{n} |A_{ij} - B_{ij}|^2 \right)^{\frac{1}{2}}$$

con $A = (A_{ij}),\ B = (B_{ij}) \in GL_n(\mathbb{R})$.

Si G es un grupo topológico, entonces las siguientes afirmaciones se cumplen:

1. La multiplicación por izquierda $\lambda_g : G \to G$, definida por $\lambda_g(x) = gx$ es un homeomorfismo.

2. La multiplicación por derecha $\rho_g : G \to G$, definida por $\rho_g(x) = xg$ es un homeomorfismo.

3. La conjugación $c_g : G \to G$, definida por $c_g(x) = gxg^{-1}$ es un homeomorfismo.

4. La inversión $inv : G \to G$, definida por $inv(x) = x^{-1}$ es un homeomorfismo.

Demostración. Estas afirmaciones nos serán muy útiles por lo que vamos a demostrar por lo menos la primera de ellas ya que la demostración misma nos será de mucha ayuda. Las demás demostraciones son análogas.

Para el (1.), fijado $g \in G$, debemos probar que λ_g es bicontinua y biyectiva. En efecto, notemos que $\lambda_g = \cdot|_{\{g\}\times G} : G \to G$, entonces la continuidad de $\cdot : G \times G \to G$ garantiza la continua de λ_g. La función $\lambda_{g^{-1}} : G \to G$ también es continua por la misma razón y además

$$(\lambda_g \circ \lambda_{g^{-1}})(x) = \lambda_g(\lambda_{g^{-1}}(x)) = \lambda_g(g^{-1}x) = gg^{-1}x = x$$
$$y$$
$$(\lambda_{g^{-1}} \circ \lambda_g)(x) = \lambda_{g^{-1}}(\lambda_g(x)) = \lambda_{g^{-1}}(gx) = g^{-1}gx = x$$

es decir,

$$\lambda_g \circ \lambda_{g^{-1}} = \mathbb{1}_G \quad \text{y} \quad \lambda_{g^{-1}} \circ \lambda_g = \mathbb{1}_G,$$

por lo que λ_g es un homeomorfismo, como queríamos. □

Dados $g, h \in G$, notemos que $\lambda_g \circ \lambda_h = \lambda_{gh}$. Sabemos ya que la multiplicación por izquierda es un homeomorfismo en el grupo G. Dicho esto, cualesquiera sean $g, h \in G$

$$\lambda_{hg^{-1}}(g) = hg^{-1}g = h$$

esto significa que cualquier par de puntos en un grupo topológico se puede conectar mediante un homeomorfismo en G, por lo tanto, todo grupo topológico es un espacio homogéneo.

Sea $\{G_\alpha \mid \alpha \in \mathscr{A}\}$ una familia de grupos topológicos, entonces el grupo producto

$$G := \prod_{\alpha\in\mathscr{A}} G_\alpha$$

es un grupo topológico con respecto a la topología producto.

Si G es un conjunto dotado de dos operaciones binarias + (suma) y $\cdot$ (multiplicación o producto), se llama *anillo* si se cumplen las siguientes propiedades:

1. (G,+) es un grupo conmutativo.

2. La multiplicación $\cdot$ es asociativa.

3. Se cumplen las propiedades distributivas del producto sobre la suma, es decir

$$a \cdot (b + c) = a \cdot b + a \cdot c \;\text{ y }\; (b + c) \cdot a = b \cdot a + c \cdot a$$

4.3. Continuidad de los programas de evaluación

Sea G un grupo topológico y τ la topología que hace que la operación de grupo y la función inversión sean continuas. De aquí en adelante supondremos que G está dotado de la topología τ que hace de G un grupo topológico.

Podemos interpretar un programa de evaluación $\Gamma_G = (\Gamma_1, \ldots, \Gamma_t)$ con $G = \langle S \rangle$ como una función de la siguiente manera. Para cada entrada $a \in S^n$ sea $(b_1, \ldots, b_t) \in G^t$ la salida

correspondiente de a respecto a Γ_G, es decir, $b_i = \Gamma_i(a)$ para $i = 1, \ldots, t$. Para cada $i \in \{1, \ldots, t\}$ definamos la función $\gamma_i : S^n \to G$ como

$$\gamma_i(a) = b_i.$$

Denotemos por $\mathcal{I} = \{\gamma_1, \ldots, \gamma_t\}$ y llamémoslo la *familia de funciones inducidas* del programa de evaluación Γ_G de G sobre S^n. De esta manera, tenemos definido también la función $\gamma : S^n \to G^t$ mediante

$$\gamma = \big(\gamma_i\big)_{i=1}^{t}$$

Así, para cada $a \in S^n$, tenemos que $\gamma(a) = \big(\gamma_i(a)\big)_{i=1}^{t}$. Notemos que para cada $j \in \{1, \ldots, t\}$ tenemos que

$$p_j \circ \gamma = \gamma_j$$

donde p_j es la función proyección en la j-ésima coordenada.

Consideremos ahora las siguientes dos funciones

$$f, g : \{1, \ldots, m_i\} \to \{1, \ldots, n\}$$

donde n y m_i son números enteros positivos e $i = 1, \ldots, t$. Estas funciones f y g nos permitirán expresar cada función inducida γ_i de una manera adecuada.

Lema 4.1. Cada función inducida γ_i es de la forma

$$\gamma_i(a) = a_{f(1)}^{\alpha_1} \cdots a_{f(m_i)}^{\alpha_{m_i}}$$

donde $a = (a_1, \ldots, a_n) \in S^n$, y $\alpha_j \in \mathbb{Z}$ para $j = 1, \ldots, m_i$.

Demostración. Probemos el lema mediante inducción en i. Para $i = 1$, tenemos las siguientes opciones

$$\gamma_1(a) = \begin{cases} a_j, & \text{o bien} \\ a_j^{-1}, & \text{o bien} \\ a_j \cdot a_k \end{cases}$$

para algún $j, k \in \{1, \ldots, n\}$. Esto es porque, de acuerdo a la Definición 3.1, la primera instrucción solo puede tomar directamente los elementos del conjunto generador S. Ahora supongamos que

$$\gamma_j(a) = b_j = a_{j,f(1)}^{\alpha_{j,f(1)}} \cdots a_{j,f(m_j)}^{\alpha_{j,f(m_j)}}$$

para cualquier $j \leq i$. Sea $\gamma_{i+1}(a) = b_{i+1}$. Si $b_{i+1} := a_j$ para algún $j \in \{1, \ldots, n\}$ ya está. Si $b_{i+1} := \Gamma_j^{-1}(a)$, entonces $b_{i+1} = b_j$ y b_{i+1} es de la forma que queríamos. Si $b_{i+1} := \Gamma_j(a) \cdot \Gamma_k(a)$, entonces

$$b_{i+1} = b_j \cdot b_k = a_{j,f(1)}^{\alpha_{j,f(1)}} \cdots a_{j,f(m_j)}^{\alpha_{j,f(m_j)}} \cdot a_{k,g(1)}^{\alpha_{k,g(1)}} \cdots a_{k,g(m_k)}^{\alpha_{k,g(m_k)}},$$

y el lema está probado. □

El siguiente teorema establece que cada función inducida γ_i es continua.

Teorema 4.1. Si $G = \langle S \rangle$ es un grupo topológico, entonces γ_i es una función continua para cada $i = 1, \ldots, t$.

Demostración. Dado que G es un grupo topológico, tenemos que la operación de grupo asociada a G es continua y la función de inversión inv también es continua. Para demostrar que γ_i es continua, fijemos $i \in \{1, \ldots, t\}$ y consideremos $a \in S^n$ cualquiera sea.

Sea $\gamma_i(a) = b_i$. Si $b_i := a_j$ para algún $j \in \{1, \ldots, t\}$, entonces $\gamma_i = p_j$ donde $p_j : S^n \to S$ es la función proyección en la j-ésima coordenada. Así, $\gamma_i(a) = a_j = p_j(a)$ es una función continua. Ahora, si $b_i := \Gamma_j^{-1}(a)$ para algún $j < i$, entonces $b_i = (\gamma_j(a))^{-1}$. Por lo tanto, $\gamma_i(a) = (\gamma_j(a))^{-1} = inv(\gamma_j(a))$ y como $\gamma_j(S^n) \subseteq G$ tenemos que $\gamma_i = inv|_{\gamma_j(S^n)}$ por lo que γ_i es una función continua por ser la restricción de una función continua.

Por último, si $b_i := \Gamma_j(a) \cdot \Gamma_k(a)$, entonces $b_i = \gamma_j(a) \cdot \gamma_k(a)$, esto significa que $\gamma_i = \cdot|_{\gamma_j(S^n) \times \gamma_k(S^n)}$, luego γ_i es una función continua. □

A continuación, probaremos que la función γ obtenida a partir de las funciones inducidas γ_i es también una función continua.

Teorema 4.2. La función $\gamma : S^n \to G^t$ definida por $\gamma = (\gamma_i)_{i=1}^t$ es una función continua.

Demostración. Las funciones proyecciones $p_j : G^t \to G$ son funciones continuas si G^t está dotado de la topología producto, tal como hemos visto en la Sección 4.1. Esto implica que para cada conjunto abierto U en G, su preimagen mediante p_j, es decir $p_j^{-1}(U)$, es un conjunto abierto en G^t; además, $p_j^{-1}(U)$ es un elemento de la subbase para la topología producto de G^t.

Para probar que γ es continua, consideremos cualquier elemento de la subbase, digamos $p_j^{-1}(U)$, y demostremos que $\gamma^{-1}(p_j^{-1}(U))$ es un conjunto abierto en S^n. De hecho

$$\begin{aligned} \gamma^{-1}(p_j^{-1}(U)) &= (\gamma^{-1} \circ p_j^{-1})(U) \\ &= (p_j \circ \gamma)^{-1}(U) \\ &= \gamma_j^{-1}(U). \end{aligned}$$

En esta última expresión, por el Teorema 4.1, sabemos que cada función inducida $\gamma_i : S^n \to G$ es continua, por lo que $\gamma_j^{-1}(U)$ es un conjunto abierto ya que U es un conjunto abierto en G. Por lo tanto, la función γ es una función continua. □

Definición 4.1. Sea Γ_G un programa de evaluación en G sobre S^n con una familia de funciones inducidas $\mathcal{I} = \{\gamma_1, \ldots, \gamma_t\}$. Decimos que un elemento $g \in G$ es computado por el programa de evaluación Γ_G si existe $a \in S^n$ tal que

$$g = \gamma_{i_0}(a)$$

para algún $i_0 \in \{1, \ldots, t\}$. Análogamente, decimos que un conjunto F es computado por el programa de evaluación Γ_G si existe $a \in S^n$ tal que

$$F \subseteq \bigcup_{i=1}^{t} \{\gamma_i(a)\}.$$

Considerando el Ejemplo 3.1, y de acuerdo a la Definición 4.1, podemos decir que Γ_{D_6} computa cualquier subconjunto del conjunto $\{\lambda,\ \rho,\ \rho^2,\ \rho^3,\ \lambda\rho^3\}$.

En el mismo sentido, del Ejemplo 3.2 concluimos que Λ_{S_6} computa cualquier subconjunto del conjunto $\{\alpha,\ \beta,\ \alpha^2,\ \alpha^2\beta,\ \alpha^2\beta\alpha,\ \alpha^2\beta\alpha\beta,\ \alpha^2\beta\alpha\beta\alpha^2\beta\alpha\beta\}$.

El siguiente teorema caracteriza los conjuntos computados por un programa de evaluación en términos de su familia de funciones inducidas.

Teorema 4.3. Si Γ_G es un programa de evaluación en G sobre S^n que computa a un conjunto F y con una familia de funciones inducidas $\mathcal{I} = \{\gamma_i\}_{i=1}^t$, entonces

$$F = \bigcup_{i \in I} \gamma_i(\gamma_i^{-1}(F)),$$

para algún $I \subseteq \{1, \ldots, t\}$.

Demostración. Dado que el programa de evaluación Γ_G computa F, entonces de acuerdo a la Definición 4.1, existe $a \in S^n$ tal que

$$F \subseteq \bigcup_{i=1}^{t} \{\gamma_i(a)\}$$

lo cual significa que para cada $y \in F$ tenemos que $y = \gamma_{i_y}(a)$ para algún $i_y \in \{1, \ldots, t\}$. Esto implica que $a \in \gamma_{i_y}^{-1}(y)$ y por ende

$$\gamma_{i_y}(a) \in \gamma_{i_y}(\gamma_{i_y}^{-1}(y)) \subseteq \gamma_{i_y}(\gamma_{i_y}^{-1}(F)).$$

Por lo tanto, para cada $y \in F$ se verifica que $y \in \gamma_{i_y}(\gamma_{i_y}^{-1}(F))$ para algún $i_y \in \{1, \ldots, t\}$. Denotemos por I al conjunto de todos los i_y en $\{1, \ldots, t\}$ que satisfacen la última afirmación. De esta manera

$$F \subseteq \bigcup_{i \in I} \gamma_i(\gamma_i^{-1}(F)).$$

Por otro lado, es fácil ver que $\gamma_i(\gamma_i^{-1}(F)) \subseteq F$ para cada $i \in \{1, \ldots, t\}$, por lo que en particular se cumple que

$$\bigcup_{i \in I} \gamma_i(\gamma_i^{-1}(F)) \subseteq F.$$

Luego, de ambas inclusiones concluimos que $F = \bigcup_{i \in I} \gamma_i(\gamma_i^{-1}(F))$, como queríamos. □

Si consideramos que G es un grupo finito, el siguiente teorema nos da un argumento cons-

tructivo de un programa de evaluación que computa la clausura de un conjunto computable. Aunque el resultado es relativamente trivial, quizá, lo importante aquí es la construcción misma del programa de evaluación.

Teorema 4.4. Si F es un conjunto computado por un programa de evaluación Γ_G con $G = \langle S \rangle$ un grupo finito, entonces la clausura de F, $\overline{F}$, en G es computable por un programa de evaluación en G de longitud al menos $L_G(F, S^n)$.

Demostración. Como F es computado por Γ_G, entonces existe $a \in S^n$ tal que

$$F \subseteq \{\gamma_1(a), \ldots, \gamma_t(a)\}.$$

Debido a que G es finito y $\overline{F} \subseteq G$, entonces $\overline{F}$ también es finito. Considerando que $F \subseteq \overline{F}$, mostraremos como construir un programa de evaluación que computa cada $x \in \overline{F} - F$.

Ya que G es un grupo topológico, sabemos que G es un espacio topológico homogéneo, por lo que para cada par de puntos $x, y \in G$ existe un homeomorfismo que envía x a y. De hecho, para cada $g \in G$, la función multiplicación por izquierda $\lambda_g : G \to G$ definida por $\lambda_g(x) = gx$ para cada $x \in G$ es un homeomorfismo. Si $g = yx^{-1}$ entonces tenemos que

$$\lambda_g(x) = gx = (yx^{-1})x = y(x^{-1}x) = y,$$

lo cual implica que $\lambda_{yx^{-1}}$ es un homeomorfismo que lleva x a y. Tal y como lo habíamos visto en la Sección 4.2.

Denotemos por $\{i_1, \ldots, i_{\bar{t}}\} \subseteq \{1, \ldots, t\}$ el conjunto de todos los índices i_j tales que $\gamma_{i_j}(a) \in F$ para cada $j = 1, \ldots, \bar{t}$ y donde t es la longitud de Γ_G. Construimos un programa de evaluación Λ_G como sigue. Si $\overline{F} - F = \{x_{i_{\bar{t}+1}}, \ldots, x_{i_{\bar{t}+r}}\}$, entonces definamos Λ_j como

$$\Lambda_j(a) := \begin{cases} \Gamma_{i_j}(a) & \text{for } j = 1, \ldots, \bar{t} \\ \lambda_{x_{i_j} \cdot \Gamma^{-1}_{i_{j-1}}(a)}(\Gamma_{i_{j-1}}(a)) & \text{for } j = \bar{t}+1, \ldots, \bar{t}+r. \end{cases}$$

El programa de evaluación $\Lambda_G = \{\Lambda_1, \ldots, \Lambda_{\bar{t}+r}\}$ computa todos los subconjuntos del conjunto

$$\{\gamma_{i_1}(a), \ldots, \gamma_{i_{\bar{t}}}(a), x_{i_{\bar{t}+1}}, \ldots, x_{i_{\bar{t}+r}}\}$$

ya que para $a \in S^n$,

$\Lambda_j(a) = \Gamma_{i_j}(a) = \gamma_{i_j}(a) \in F$ para $j = 1, \ldots, \bar{t}$

y

$\Lambda_j(a) = \lambda_{x_{i_j} \cdot \Gamma^{-1}_{i_{j-1}}(a)}(\Gamma_{i_{j-1}}(a)) = x_{i_j} \cdot \Gamma^{-1}_{i_{j-1}}(a) \cdot \Gamma_{i_{j-1}}(a) = x_{i_j} \in \overline{F} - F$ para $j = \bar{t}+1, \ldots, \bar{t}+r$

Por lo tanto, Λ_G computa $\overline{F}$ y su longitud es $\bar{t} + r \geq t \geq L_G(F, S^n)$. □

El siguiente teorema afirma que todo conjunto computado por un programa de evaluación

en G es de hecho un conjunto compacto en G.

Teorema 4.5. Si F es un conjunto computado por un programa de evaluación Γ_G, entonces F es un subconjunto compacto de G.

Demostración. Sea $\mathcal{U}$ un cubrimiento abierto de F en G. Como F es computado por Γ_G, entonces F es un conjunto finito, por lo tanto, para cada $y \in F$ escogemos un entorno U_y en $\mathcal{U}$, lo cual implica que

$$F \subseteq \bigcup_{y \in F} U_y.$$

Esto significa que $\mathcal{U}$ contiene una subfamilia finita que también cubre a F. Por lo tanto F es un subconjunto compacto de G. □

4.4. Familia de funciones inducidas en anillos

Ahora consideramos un anillo $(G, \cdot, +)$ donde $(G, \cdot)$ es un grupo topológico, $(G_0, \cdot)$ es un monoide topológico con $G_0 = G \cup \{0\}$ y 0 es el único elemento de G_0 sin inversa.

Definición 4.2. Si $\mathcal{I} = \{\gamma_1, \ldots, \gamma_t\}$ es una familia de funciones inducidas de algún programa de evaluación Γ_G, definimos una función $\mu : G_0 \to G_0$ sobre el anillo $(G_0, \cdot, +)$, con 0 como la identidad aditiva, mediante

$$\mu(a) = \sum_{i=1}^{t} \gamma_i(a).$$

Llamaremos a esta función μ la *función generada* de Γ_G.

Del Teorema 4.1, sabemos que cada función inducida γ_i es continua, por lo tanto, el siguiente lema resulta de inmediato.

Lema 4.2. Si $(G, +)$ es un grupo topológico, entonces la función generada μ por un programa de evaluación Γ_G es continua.

Demostración. Si $(G, +)$ es un grupo topológico, entonces la función operación de grupo $+ : G \times G \to G$ es continua, por lo que μ es continua ya que cada γ_i también lo es. □

Si la función μ es un polinomio, entonces podemos acotar su grado, según indica el siguiente teorema al respecto.

Teorema 4.6. Si μ es un polinomio generado por algún programa de evaluación Γ_G de longitud t, entonces el grado de μ es al menos 0 y a lo sumo 2^{t-1}.

Demostración. Si μ es un polinomio, entonces cada función γ_i la podemos considerar como un monomio. Denotemos por $deg(p)$ el grado de un polinomio p.

Si $\gamma_i(a) = \Gamma_i(a) = a_j$, entonces $deg(\gamma_i) = 1$; si $\gamma_i(a) = \Gamma_i(a) = \Gamma_j^{-1}(a)$ para algún $j < i$, entonces $deg(\gamma_i) \geq 1$; si $\gamma_i(a) = \Gamma_i(a) = \Gamma_j(a) \cdot \Gamma_k(a)$, entonces $deg(\gamma_i) \geq 0$.

Como $\mu(a) = \gamma_1(a) + \cdots + \gamma_t(a)$, entonces tenemos que $deg(\mu) \geq deg(\gamma_i) \geq 0$. Por otro lado, para determinar el mayor grado posible de μ, tomemos cualquier $a_i \in \{a_1, \ldots, a_n\} \subseteq S$ y multipliquémoslo consigo mismo tantas veces como podamos. Para este efecto, podemos construir un programa de evaluación que haga eso, es decir, existe un programa de evaluación tal que

$$\begin{aligned} b_1 &= \gamma_1(a) = a_i, \\ b_2 &= \gamma_2(a) = \gamma_1(a) \cdot \gamma_1(a) = a_i^2, \\ &\vdots \\ b_t &= \gamma_t(a) = \gamma_{t-1}(a) \cdot \gamma_{t-1}(a) = a_i^{2^{t-1}}, \end{aligned}$$

y por lo tanto, $deg(\mu) \leq 2^{t-1}$. De esta manera, demostramos que $0 \leq deg(\mu) \leq 2^{t-1}$

□

Por último, como μ es la suma de las funciones inducidas γ_i, es lógico pensar que si tenemos dos programas de evaluación Γ_G y Δ_G que generan dos funciones, entonces la suma de estas funciones generadas vuelve a ser una función generada de algún programa de evaluación en G. El siguiente teorema nos habla justamente al respecto. Como hemos mencionado en

Teorema 4.7. Sean μ y ρ dos funciones generadas por los programas de evaluación Γ_G y Δ_G, respectivamente. Entonces la función $\lambda : G_0 \to G_0$ definida como $\lambda = \mu + \rho$ es una función generada de algún programa de evaluación en G.

Demostración. Dado que μ y ρ son funciones generadas de los programas de evaluación Γ_G y Δ_G, respectivamente, tenemos que

$$\mu(a) = \sum_{i=1}^{t} \gamma_i(a) \quad \text{y} \quad \rho(b) = \sum_{i=1}^{r} \delta_i(b),$$

donde $\big(\gamma_i\big)_{i=1}^{t}$ y $\big(\delta_i\big)_{i=1}^{r}$ son familias de funciones inducidas de Γ_G y Δ_G, respectivamente, con $a \in S^n$, $b \in R^m$, siendo $G = \langle S \rangle = \langle R \rangle$.

Podemos definir una nueva función de la siguiente manera

$$\lambda(ab) = \mu(a) + \rho(b) = \sum_{i=1}^{t+r} \varphi_i(ab),$$

donde

$$\varphi_i(ab) = \begin{cases} \gamma_i(a), & \text{para } i \leq r \\ \delta_{i-r}(b), & \text{para } r < i \leq t + r. \end{cases}$$

Esto produce un programa de evaluación $\Lambda_G = (\Lambda_1, \ldots, \Lambda_{t+r})$ que en la entrada $\big((S \cup R)^{n+m}, ab\big)$ genera la función λ.

□

CAPÍTULO 5
CONCLUSIÓN

En este trabajo iniciamos el estudio de los programas de evaluación definidos sobre grupos. Estudiamos la complejidad computacional de un programa de evaluación que computa subconjuntos de un grupo y sus propiedades topológicas siempre que el grupo sea un grupo topológico.

Uno de nuestros principales resultados es una técnica de *simulación* que nos permite realizar el estudio que nos hemos planteado, comparar la complejidad conmutativa contra la complejidad no conmutativa. Específicamente nos permite comparar la longitud de un programa de evaluación definido sobre un grupo contra la longitud de otro programa de evaluación definido sobre un grupo (potencialmente diferente). Este resultado está expresado en el Teorema 3.1, el cual establece que $\mathcal{L}_{nc}(\hat{F}, n) \leq \mathcal{L}_c(F, n)$ lo cual significa que la complejidad no conmutativa de la extensión isomorfa de un conjunto F es menor o igual a la complejidad conmutativa del conjunto F. Tal y como los hemos explicitado en el Capítulo 1, esto va en contra de nuestra intuición, pero esto se debe justamente a la falta de cancelación en las estructuras algebraicas no conmutativas.

A parte de esta técnica de simulación que nos permite relacionar las complejidades entre diferentes familias de grupos, establecemos las bases teóricas para su utilización e introducimos el concepto de extensión isomorfa que nos ayuda justamente a poder relacionar dos grupos entre sí y por ende, comparar las complejidades.

El hecho que un programa de evaluación simule otro programa de evaluación, establece una relación en el conjunto de los programas de evaluación que computan un conjunto dado, salvo un isomorfismo. En este sentido, también estudiamos las propiedades de esta relación de simulación en dicho conjunto, y además estudiamos algunas características interesantes de la simulación.

Otro resultado relevante es que siempre que se considere un programa de evaluación defi-

nido sobre un grupo topológico, cada programa de evaluación define una familia de funciones inducidas que cumplen varias propiedades interesantes. Una de ellas es que cada instrucción del programa de evaluación induce funciones continuas. Además, los conjuntos computados por el programa de evaluación son caracterizados por esta familia de funciones inducidas del programa de evaluación (Teorema 4.3).

Existe mucho por investigar desde esta perspectiva, de seguro. A continuación citamos algunos problemas abiertos y líneas de investigación que consideramos pueden mejorar nuestra comprensión sobre los programas de evaluación definidos sobre estructuras de grupos, y más aún, si esta estructura algebraica tiene una estructura topológica asociada.

1. *Una separación estricta ($<$) entre la complejidad de evaluación conmutativa y no conmutativa.*

   En el Lema 3.5 y el Teorema 3.1 mostramos que la longitud de cualquier programa de evaluación en un grupo conmutativo es una cota superior para la longitud de algún programa de evaluación en un grupo no conmutativo. Lo que queda por hacer es presentar una construcción concreta de un conjunto que haga que las desigualdades del Lema 3.5 y del Teorema 3.1 sean en un sentido estricto.

2. *Condiciones para la existencia de extensiones isomorfas.*

   En Definición 3.4 presentamos la definición de extensión isomorfa, el cual nos permite construir programas de evaluación que simulan otros programas de evaluación. Falta investigar cuales son las condiciones suficientes o necesarias para la existencia de tales extensiones isomorfas, por ejemplo, basadas en alguna invariante topológica.

3. *Invariantes topológicos que se preservan mediante la simulación de programas de evaluación.*

   Muy relacionado con el ítem anterior, se puede estudiar qué invariantes topológicos se preservan cada vez que un programa de evaluación en un grupo es simulado por otro programa de evaluación en un grupo distinto. La proposición 4.5 muestra un primer paso hacia esta línea de investigación.

4. *Conexidad (conexidad por caminos) de las familias de funciones inducidas.*

   Una pregunta interesante es si para cada $a \in S^n$ la familia de funciones inducidas $\{\gamma_1, \ldots, \gamma_n\}$ es conexa (conexa por caminos) o no en G. Esto implicaría que los conjuntos computados por diferentes programas de evaluación están en diferentes componentes conexas (conexas por caminos) dentro de G.

5. *Factorización de las funciones generadas.*

   En el Teorema 4.7 mostramos como se puede obtener una nueva función generada a partir de la suma de dos funciones generadas por programas de evaluaciones dadas previamente. Una pregunta interesante es si podemos obtener la función inversa, por ejemplo, dada un

función μ se puede estudiar si podemos obtener dos funciones generadas que sumadas den la función μ.

REFERENCIAS

Nisan, N. (1991). Lower bounds for non-commutative computation. *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, 410-418.

Bürgisser, P., Clausen, M., & Shokrollahi, M. A. (2013). *Algebraic complexity theory* (Vol. 315). Springer Science & Business Media.

Raz, R., & Shpilka, A. (2005). Deterministic polynomial identity testing in non-commutative models. *computational complexity*, *14*(1), 1-19.

Arvind, V., Joglekar, P. S., & Raja, S. (2016). Noncommutative valiant's classes: Structure and complete problems. *ACM Transactions on Computation Theory (ToCT)*, *9*(1), 1-29.

Arvind, V. (2013). Noncommutative Arithmetic Circuits meet Finite Automata. *Bulletin of EATCS*, *2*(104).

Chien, S., & Sinclair, A. (2004). Algebras with polynomial identities and computing the determinant. *45th Annual IEEE Symposium on Foundations of Computer Science*, 352-361.

Babai, L., & Szemerédi, E. (1984). On the complexity of matrix group problems I. *25th Annual Symposium onFoundations of Computer Science, 1984.*, 229-240.

Nies, A., & Tent, K. (2017). Describing finite groups by short first-order sentences. *Israel Journal of Mathematics*, *221*(1), 85-115.

Shpilka, A., & Yehudayoff, A. (2010). Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, *5*(3–4), 207-388.

Lynch, N. A. (1980). Straight-line program length as a parameter for complexity analysis. *Journal of Computer and System Sciences*, *21*(3), 251-280.

Alonso, C. L., Puente, J., & Montana, J. L. (2008). Straight line programs: a new linear genetic programming approach. *2008 20th IEEE International Conference on Tools with Artificial Intelligence*, *2*, 517-524.

Giusti, M., Heintz, J., Morais, J. E., Morgenstem, J., & Pardo, L. M. (1998). Straight-line programs in geometric elimination theory. *Journal of pure and applied algebra*, *124*(1-3), 101-146.

Bank, B., Giusti, M., Heintz, J., & Mbakop, G. M. (1996). Polar varieties, real equation solving and data-structures: the hypersurface case. *arXiv preprint alg-geom/9609004*.

Kranz, T., Leander, G., Stoffelen, K., & Wiemer, F. (2017). Shorter linear straight-line programs for MDS matrices. *IACR Transactions on Symmetric Cryptology*, 188-211.

Paterson, M. S., & Stockmeyer, L. J. (1973). On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, *2*(1), 60-66.

Strassen, V. (1974). Polynomials with rational coefficients which are hard to compute. *SIAM Journal on Computing*, *3*(2), 128-149.

Heintz, J. (1986). On polynomials with symmetric Galois group which are easy to compute. *Theoretical computer science*, *47*, 99-105.

von zur Gathen, J. (1985). Irreducibility of multivariate polynomials. *Journal of Computer and System Sciences*, *31*(2), 225-264.

Ibarra, O. H., & Moran, S. (1983). Probabilistic algorithms for deciding equivalence of straight-line programs. *Journal of the ACM (JACM)*, *30*(1), 217-228.

Kaltofen, E. (1988). Greatest common divisors of polynomials given by straight-line programs. *Journal of the ACM (JACM)*, *35*(1), 231-264.

Kaltofen, E. (1989). Factorization of polynomials given by straight-line programs. *Adv. Comput. Res.*, *5*, 375-412.

Schleimer, S. (2008). Polynomial-time word problems. *Commentarii mathematici helvetici*, *83*(4), 741-765.

Artin, M. (1991). *Algebra*. Prrentice-Hall, Inc.

Dorronsoro, J., & Hernández, E. (1996). *Números, grupos y anillos*.

Krick, T. (2002). Straight-line programs in polynomial equation solving. *Foundations of computational mathematics: Minneapolis*, *312*, 96-136.

Arora, S., & Barak, B. (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Dugundji, J. (1966). *Topology* (12.ª ed.). Allyn; Bacon, Inc.

Munkres, J. R. (2000). *Topology* (Vol. 2). Prentice Hall Upper Saddle River.

McCarty, G. (1988). *Topology: an introduction with application to topological groups*. Courier Corporation.